ICT375Ans1

Client side programming technologies are used to build web pages an application that run on the client such as browsers on users device. Often referred to as front end

Server side programming involves the application that respond to requests from client side web browsers. Often referred to as back end

httpd.conf file tells the server how to run it is read first when Apache is started up.

.htaccess file controls directory options such as permissions. Must be enabled by httpd.conf

ICT375Ans2

https://www.reddit.com/r/javascript/comments/5m6tkz/should_i_use_es6_classes_or_not_js_beginner/ ← Very good

Javascript arguments:

- The premise is that function parameter always has a single array object (argument object is the correct name not array) and the function parameters are considered elements of the array object. OR Empty array
- Javascript functions do not require any parameters to be stated
    - arguments[i] is the reference to function parameters
- A function can be called without the calling function having the same number of parameters as function parameters

- NOTE: AVOID DUE TO READABILITY. Also need to keep track of index and arguments is not descriptive enough

```
function myFunc(fName) {
        console.log(arguments[0] OR fName, arguments[1], arguments[2]);
}
myFunc("First name", "Last Name", 30);
```

Anonymous functions + High order functions:

- Write Javascript function without direct name
- Used for when we want a variable to store a function!

- For: function will be passed into other parameters (higher ordered functions)
    - Ask yourself- Why name the function if we are never going to refer to it?
- For: when javascript function is ever going to be called in one place but bad for readability and reusability

```
var storeFunction = function () {

      …

}

SetTimeOut(storeFunction, 3000);

SetTimeOut(function () { …;}, 3000);
```

Closure functionality + Lexical environment:

- Inner functions have access to all the outer functions scope (includes variables and function calls)!
- The lexical environment of inner function contains variables, and function calls of outer function scope at which the inner function was created

```
function foo(arg1, arg2) {

    var var1 = arg1;
    var var2 = arg2;
    //console.log("SHOWS FUNCTION CALL IS ALSO PART OF LEXICAL ENVIRONMENT"); //SELF NOTE: Why add this? Shows lexi environment consist of this function and it is called

    function bar() { //SELF NOTE: Why not return function bar() ? Because this is for readability and understanding that we return function not bar() calling
        var sum = arg1 + arg2;
        console.log("Sum of parameters: ", sum);

    }
    return bar;
}

var foo2 = foo(3,5); //SELF NOTE: Why not foo(3,5) only? Doesn't work since think currently var foo2 will store bar function. When foo2 is called bar is executed
foo2();            //W/O var foo2 = ... then foo is called and bar function is returned but not executed
```

Callback functions:

- Pass a generic function a parameter and call that function
- Used for when function needs to execute one different functions depending on context/situation. Rather than have a bunch of if statements just have a general function in its parameter and call it

Example of using callback functions:

Use of callback

The problem-

```
function reqSolarRadiation(request, response) {
	LoadJSONFromWebsite(fullURLPath);
}

function reqWindSpeed(request, response) {
	LoadJSONFromWebsite(fullURLPath);
}

function LoadJSONFromWebsite(fullURLPath) {

	http.get(fullURLPath, function (data) {  //NOTE: Can't return in asynchronous function
		//ISSUE- How do we know when to execute GetWindspeed() or GetSolarRadiation() function?
		//Poor solution- Create LoadJSONFromWebsiteWS(…) + LoadJSONFromWebsiteSR(..) function (duplication)
		//Poor solution- Pass a variable parameter to LoadJSONFromWebsite(..) and use if statements to direct
	}
}
```

My ideal solution using callback-

```
function reqSolarRadiation(request, response) {
      LoadJSONFromWebsite(fullURLPath, GetSolarRadiation);
}

function reqWindSpeed(request, response) {
      LoadJSONFromWebsite(fullURLPath, GetWindSpeed);
}

function LoadJSONFromWebsite(fullURLPath, CallSpecifiedFunction) {

      http.get(fullURLPath, function (data) {
            CallSpecifiedFunction();
      }
}
```

Thus, no messy if-statements inside http.get(…) and only one LoadJSONFromWebsite(…) function needed. And the notion you can't return http.get(..) is followed

Javascript object creation methods-

Object default constructor:

```
var Person = new Object(); //MUST BE Always Object()
Person.property = 40;
Person.print = function () { //ANONOYMOUS FUNCTION
      console.log("This is object construction");
}
console.log(Person[“property”]); //Square brack notation method
Person.Print();
```

Object literals:

```
var Student = {
      property : "Jin",
      property2 : 41
};
console.log("Age: ", Student[“property2”]);
```

ES6 class:

```
class ClassName {
      constructor(newAge) { //MUST BE Always constructor()
            this.age = newAge;
      }

      display() {
            console.log("Age: ", this.age);
      }
```

```
}
var student = new ClassName(22);
student.display();
```

Asynchronous functions:

- Where there is no waiting for server response for client request
- Separate thread will send request to server and wait for response and call call-back function
- Ajax resolves this issue and client-side response without blockage client side
- BUT server side has blockage such as searching database, so solution is use Node.JS (thus asynchronous)
- Apache uses child processes not threads thus very performance intensive

- Callback function is what happens when query is done instead of waiting and holding everything up

NOTES:

- Variables defined outside Asynchonrous functions can be read inside the function because they are treated as global
    - https://stackoverflow.com/questions/1904376/in-jquery-post-how-do-i-get-value-of-variable-outside-function
- The function containing asynchonour functions will process straight through before THE async function is complete thus assume variables outside asynchronous function and changed inside asynconour function doesn't get changed
- Treat asynchronous function as one way meaning all it does it execute doesn't return to outer function

Node.JS:

- Used to make server-side web applications
- Difference with PHP file is that it is a process (terminal) running in the background that gets clients HTTP request.
- Must setup what happens (how to handle) when HTTP request is sent to server (like user enters URL link)
- Node JS web server is like Apache web server (general purpose high overhead)
- Contains event loop that looks for event and passes to callback

- NOTE: Can't put nodeJS and front end Javascipt into a single file. Because when you run nodeJS script it can't run browser/html stuff. Because when you run vanilla JS it can't executed nodeJS stuff (modules)
    - `Window.closed` ← `Put in nodeJS it will not run`
    - `document.querySelector('body')` ← `nodeJS file`
    - `require('fs')` ← `Vanilla JS file`

HTTP: responsible for Node.JS web server

Method	Return	

ICT375Ans3

Socket can be thought of as a wall socket it allows the TCP and UDP to connect between two network programs. Both UDP and TCP allow communication and retrieving messages. Programmers define socket connections which include port number. From port number we know which socket gets this message (not relevant in this unit

Get method is for requesting a specific resource (server)

Post method is for submitted data to be processed to the specified resource usually to modify existing resource (server) (POST can be used in replacement for Get method)

Put method uploading/overwriting a specific resource to server

NodeJS exporting and important importance:

- Allows nodeJS files to use other nodeJS files with multiple methods

NodeJS Basic HTTP Web Client (…)

- For testing purposes only

NodeJS Basic HTTP Web Server (server.js)

- HTTP server waits for request and then server calls user-defined call back function
- All we need to do is register call back function which means we tell server what it should do when request is received
- Can communicate with HTTP client using GET, HEAD and Post
- Functions comes from HTTP module (core module of nodeJS)

```
"use strict";

//Import modules
var http = require('http');
var url = require('url');

//Export HTTP server
function startServer(route /*,handle*/) { //SELF NOTE: Why not startServer()? Because we want access to router.js- route method
    //Creates the HTTP web server
    http.createServer( function (request, response) { //SELF NOTE: Why not provide function name? Because in this file we want to define + call server in one method
        //var urlQueryString = url.parse(request.url).query; //SELF NOTE: Why not url.parse(response.url)? Because client sent/called http.Request from ex4c.js and server sends respons
        var pathName = url.parse(request.url).pathname;
        route(pathName, handle); //SELF NOTE: Why not just route(PathName/request.url)? Because question requires urlQueryString and by convention send pathName string not request.url

        response.writeHead(200,{'Content-Type': 'text/plain'}); //SELF NOTE: Why not request.write(...)? Because client sent/called http.Request(...) server sends response e.g response
        response.write('Server has been started');
        response.end();
    }).listen(40004, 'ceto.murdoch.edu.au') //This will call the createServer method when server is accessed by client

    console.log('Server running http://ceto.murdoch.edu.au:40004/');
    console.log('Proces ID: ', process.pid);
}

exports.startServer = startServer; //SELF NOTE: Why export server? Because this is so other nodeJS can use all of server-ex5.js methods
```

- Responsible for passing URL to routing method (router.js)

Routing (router.JS)

- Responsible for pointing/directing program flow to which request handler method (requestHandler.JS) to call when given a specific HTTP request (from server.JS)
    - Both ways request and response

Index.JS

- Run this file first to start the server and others modules
- Is the starting point for application contains- server creation and routing

JSON.Encode/Decode

[Insert format]

… (request object, response object)

… (index.js)

… (router.js)

Direct to request handlers

ICT375Ans4

Request Handler (requestHandler.js)

- Contains request handler methods and is responsible for executing the client request and response given a specific HTTP request
- Responsible for reading request + generating response + send response to client rather than scattered in server as well

Server Responding to Client Request-

- Pass the Response Object → Router → requestHandler (Response.Write) deals with response
- Shift responsibility of writing a response (response.write) to the requestHandler.js

Display HTML:

- For searching use JQuery Post → JSON → Display HTML (JSON)
- For form input use Form Post → Event handler → Create HTML Backend

Uploading image file

display image file-

…(requestHandler.js)

Do something in charge of executing

ICT375Ans5

XML

- A markup language like HTML provides information about structure and content of document by programs
- Store vast amount of data

- Ability to create new mark-up language/vocab (think ICT365) for different context
- So, documents are made using an XML language (language that is defined using XML)

- Applications of XML → XHTML and Math Markup Language (MathML)

Document Type Definition (DTD):

- One way to define the structure of a new XML documents/language
- Outline which tags are only allowed, which tags contain other tags, and location of basic text data

Well-formed XML document contains:

- Prolog (headers) + XML declaration (must) + DTD (must)
- Root element
- Miscellaneous parts

Namespaces allow for use of > 1 DTD

ICT375Ans6

XML Schema:

- Is alternative method to DTD it provides a way to define the structure of a new XML documents/language
- Written in XML
- Outline which tags are only allowed, which tags contain other tags, and location of basic text data

- DTD syntax to define element and attributes != XML document syntax to define element and attributes

ICT375Ans7

Parse and process XML documents

Parsing is known as syntactical analysis. Positioned between xml application and xml documents.

Parser is an application that processes information on XML document

Process: occurs when XML document is parsed it can then extract the relevant data in XML document (parse object)

ICT375Ans8

JSON string/text is the universal type that works for all languages and is lightweight

JSON.parse(…) takes the JSON string and converts it into native type/object of the language (Javascipt)